1. 請問buffersize分別是：0、-1、4KB、16KB、64KB、1MB、8MB的執行速度分別為何？（使用time指令）

```
mysp@SP:~/system-programming/ch05/hw04$ time ./fileperf input.txt outt.txt -1

real    0m2.506s
user    0m0.510s
sys     0m0.707s
mysp@SP:~/system-programming/ch05/hw04$ time ./fileperf input.txt outt.txt 0

real    0m31.250s
user    0m5.150s
sys     0m10.408s
mysp@SP:~/system-programming/ch05/hw04$ time ./fileperf input.txt outt.txt 4096

real    0m0.758s
user    0m0.359s
sys     0m0.017s
mysp@SP:~/system-programming/ch05/hw04$ time ./fileperf input.txt outt.txt 16384

real    0m0.746s
user    0m0.362s
sys     0m0.010s
mysp@SP:~/system-programming/ch05/hw04$ time ./fileperf input.txt outt.txt 65536

real    0m0.736s
user    0m0.351s
sys     0m0.013s
mysp@SP:~/system-programming/ch05/hw04$ time ./fileperf input.txt outt.txt 1048576

real    0m0.724s
user    0m0.343s
sys     0m0.020s
mysp@SP:~/system-programming/ch05/hw04$ time ./fileperf input.txt outt.txt 8388608

real    0m0.697s
user    0m0.331s
sys     0m0.020s
```

如圖所示:
當buffersize設為0、-1、4KB、16KB、64KB、1MB、8MB所需執行時間
可以發現unbuffered的情況下執行最慢(31.250s)
又隨著buffersize越大執行時間越短，然過一個臨界點後bufsize的變大對效能的影響趨緩

2.使用ltrace觀察你的應用程式呼叫「函數庫的情況」

```
mysp@SP:~/system-programming/ch05/hw04$ ltrace -c ./fileperf input.txt outt.txt 4096
% time     seconds  usecs/call     calls      function
------ ----------- ----------- --------- --------------------
 23.35    0.000156         156         1 fopen
 15.72    0.000105         105         1 setvbuf
 13.77    0.000092          92         1 malloc
 13.02    0.000087          87         1 __isoc99_sscanf
 10.33    0.000069          69         1 getc
  8.83    0.000059          59         1 clock_gettime
  7.49    0.000050          50         1 fputs
  7.49    0.000050          50         1 fputc
------ ----------- ----------- --------- --------------------
100.00    0.000668                    8 total
mysp@SP:~/system-programming/ch05/hw04$ ltrace ./fileperf input.txt outt.txt 4096
fopen("input.txt", "r")                                          = 0x5607144c32a0
__isoc99_sscanf(0x7ffda917c1db, 0x5607133da009, 0x7ffda917ac34, 8) = 1
setvbuf(0x5607144c32a0, 0, 0, 4096)                              = 0
malloc(4096)                                                     = 0x5607144c4670
clock_gettime(1, 0x7ffda917ac60, 0, 3072)                        = 0
getc(0x5607144c32a0, 0x7ffda91f9090, 0, 24)                      = 67
fputs("Computer", 0x5607144c3480)                                = 1
fputc('\n', 0x5607144c3480)                                      = 10
+++ exited (status 0) +++
```

指令ltrace ./fileperf input.txt outt.txt 4096
由圖可見這個程式呼叫了那些函數庫、記憶體使用(fopen、malloc)、函數回傳值
(使用-c則可表單式觀察每個函式使用次數、占比及時間)

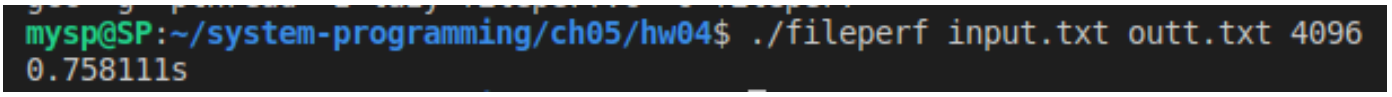
3.使用strace觀察你的應用程式呼叫「作業系統的情況」

```
mysp@SP:~/system-programming/ch05/hw04$ strace -c ./fileperf input.txt outt.txt 4096
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 72.91    0.004454           1      3307           write
 27.09    0.001655           0      3347           read
  0.00    0.000000           0         2           close
  0.00    0.000000           0         3           fstat
  0.00    0.000000           0         7           mmap
  0.00    0.000000           0         4           mprotect
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         6           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         2         1 arch_prctl
  0.00    0.000000           0         4           openat
------ ----------- ----------- --------- --------- ----------------
100.00    0.006109                  6688         2 total
```

```
write(4, "If\00032_PL0_SSP\0H_STK_EN\0its set. "..., 4096) = 4096
read(3, "xed at 8+2 bytes. The instructio"..., 4096) = 4096
write(4, " generated\0n\0stem\0[bit\0s_Until(p"..., 4096) = 4096
read(3, "\nA_1 \357\203\237 f (B, C, D) + (A ROL 5)"..., 4096) = 4096
write(4, " four\0rms\0\0\0\0\0\0m\0\0MIP = 0. See L"..., 4096) = 4096
read(3, "5:64] ;\nW14 \357\203\237 SRC2[63: 32] ;\nW"..., 4096) = 4096
write(4, "Instruction\0int\0u32(__m128i,\0\0ra"..., 4096) = 4096
read(3, " intermediate calculation for th"..., 4096) = 4096
```

指令 strace ./fileperf input.txt outt.txt 4096
由圖可見這個程式呼叫作業系統的狀況與buffer大小(4096)(下圖)
加上-c可表單式觀察syscall有哪些等等(上圖，同ltrace呈現的功能)，另外可由calls的次數回推buffe大略大小
例如此圖:
輸出檔案大小(13544023bytes)/3307=4095.561...

4.有辦法根據2和3分析一下「呼叫作業系統核心函數（system call）」和「函數庫呼叫」的「成本」差異嗎？



於迴圈前後加入clock_gettime()計算時間，可得知所經過時間為0.758111s
從strace與ltrace結果來看，ltrace每個函式僅呼叫一次，相比strace不斷重複呼叫使得時間多很多
從clock_gettime()結果來看，幾乎與time指令出來結果差不多，可見大部分時間佔在讀取與寫入的時候

請教的朋友: 郭怡靚、論壇上回答我問題的老師和ray1422